Temporal Task Allocation in Periodic Environments

An Approach Based on Synchronization

Manuel Castillo-Cagigal , Arne Brutschy , Alvaro Gutiérrez , and Mauro Birattari

Abstract. In this paper, we study a robot swarm that has to perform task allocation in an environment that features periodic properties. In this environment, tasks appear in different areas following periodic temporal patterns. The swarm has to reallocate its workforce periodically, performing a *temporal task allocation* that must be synchronized with the environment to be effective.

We tackle temporal task allocation using methods and concepts that we borrow from the signal processing literature. In particular, we propose a distributed temporal task allocation algorithm that synchronizes robots of the swarm with the environment and with each other. In this algorithm, robots use only local information and a simple visual communication protocol based on light blinking. Our results show that a robot swarm that uses the proposed temporal task allocation algorithm performs considerably more tasks than a swarm that uses a greedy algorithm.

1 Introduction

In dynamical environments, real-time resource allocation commonly involves situations in which events occur periodically, with a certain frequency [14]. Periodicity can originate from both natural and artificial phenomena, for example, earth's rotation and revolution, tides, cyclic production processes, and customer demands. In artificial systems, the designer typically wishes to allocate resources so as to increase the system performance and achieve predefined goals [11]. To this end, it is paramount that information on the nature of the periodic events involved is available during the design process [9].

Task allocation as studied in swarm robotics [5] is a class of resource allocation problems: the workforce of the swarm can be seen as the resource to be allocated—see [2] for a recent review of the swarm robotics literature including works on task allocation. In this paper, we study a case in which a robot swarm needs to perform task allocation in an environment that features periodic properties. Specifically, the periodicity of the environment lies in the temporal pattern

in which new tasks appear. To operate effectively, the swarm needs to reallocate its workforce according to the periodicity of the environment. We call *temporal task allocation* a task allocation that takes into account temporal properties of the environment.

To exploit environments with periodic properties, a task allocation algorithm needs to adapt to the periodicity of the environment. In this paper, we propose a novel temporal task allocation algorithm that adapts to the environment. This algorithm is based on concepts that we borrow from the signal processing and collective synchronization literature.

Collective synchronization has been previously observed and studied in biological systems (e.g., [6]). In these systems, the components converge to a common phase and oscillate in unison. Collective synchronization is usually modeled via coupled oscillators [15]. A model that is commonly adopted is the one proposed by Kuramoto [8]. A direct application of Kuramoto's model in swarm robotics is not appropriate because it would require that each robot knows with which phase the others oscillate. Other models exist that do not require that a robots knows the phase of the others. Examples are models based on firefly synchronization and chorusing mechanisms [4,7], which are commonly based on local communication.

In this paper, we propose a temporal task allocation algorithm in which robots synchronize with each other and with the environment. The synchronization with the environment is the novelty of our work.

2 Environment and Robots

We consider a rectangular environment that is divided in three areas: workspace A, workspace B, and a transition area. Fig. 1a shows a schematic representation of the environment. Tasks appear either in workspace A or B, following a temporal pattern. Robots have to travel from workspace to workspace to attend tasks where they appear. The workspaces are separated by the transition area: a robot that moves from one workspace to the other has to cross the transition area. The time spent by the robot i to cross the transition area is called switching cost ξ_s^i. It is measured in time units and is independent for each robot. Due to possible collisions with other robots, ξ_s^i is a random variable.

We use an abstracted representation of tasks: to carry out a task, a robot has to reach the location at which the task appeared and stay there for a certain amount of time. The time ξ_e that a robot spends working on a task is fixed. Tasks have a life time ξ_l after which they expire: if a task remains unattended for longer than ξ_l, it is removed from the environment. At time k, $N^A[k]$ and $N^B[k]$ are the number of tasks present in workspace A and B, respectively. The amount of tasks in each workspace is bounded by the task capacity Γ, which is the same for the two workspaces.

The periodicity of the environment that we consider in this paper lies in the temporal pattern with which tasks appear. During a period of time T^A, new tasks appear in workspace A. After the end of T^A, new tasks appear in workspace B

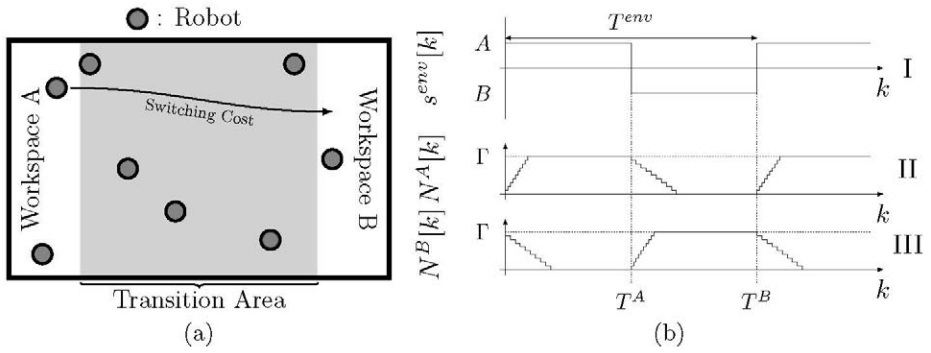

Fig. 1. Environment definition. a) Schematic representation of the arena, with workspaces A and B in white and transition area in gray. b) Environment period and location of the task appearance example: I) signal $s^{env}[k]$ of task appearance with period T^{env}, II) number $N^A[k]$ of tasks in workspace A, III) number $N^B[k]$ of tasks in workspace B.

for a period of time T^B. After the completion of T^B, new tasks appear again in workspace A, and so on. The full cycle has a period $T^{env} = T^A + T^B$. In this paper, we assume $T^A = T^B$. The location of the appearance of tasks in the environment can be described as a square signal denoted by $s^{env}[k]$ that takes a value of A or B. An example of T^{env} and $s^{env}[k]$ is shown in Fig. 1b-I.

Regardless of the workspace, the tasks appear in the environment with a certain incoming task rate λ. If the task capacity Γ of a workspace is reached, additional tasks are dismissed. When tasks no longer appear in a workspace, the number of tasks in this workspace decreases as tasks expire. This effect can be observed in Fig. 1b-II and 1b-III for both workspaces: the number of tasks increases until Γ is reached and decreases after new tasks cease to appear.

The robots move in the arena between workspace A and B in order to attend to the tasks. Robots act independently of each other, but are able to exchange simple messages via short-range line-of-sight communication. The number of robots in a workspace are the workforce allocated to this workspace by the swarm. In order to maximize performance, the swarm needs to allocate its complete workforce to the workspace where tasks are available. To achieve this goal, the robots need to switch between workspaces so that their movement is synchronized with the temporal pattern of task appearance, performing a temporal task allocation.

3 Collective Synchronization Algorithm

In this section, we present the *collective synchronization* (CS) algorithm. The goal of CS is to synchronize the movement of the robots between workspaces with the appearance of tasks in the environment. In CS, each robot i has an internal timer τ^i that governs its transitions between workspaces. This timer

increases each time step and resets to zero when it reaches the period T^i of robot i. The robot switches between workspaces depending on τ^i:

$$s^i[k] = \begin{cases} A & \tau^i \leq T^i/2 \\ B & \tau^i > T^i/2 \end{cases} \quad (1)$$

This equation produces a square signal $s^i[k]$ as shown in Fig. 2a-I. The timer τ^i might not be synchronized with the appearance of tasks in the environment. The difference between τ^i and task appearance is $\bar{\tau}^i$, defined in the range $[-T^i/2, T^i/2]$.

CS achieves synchronization in two steps. First, each robot i evaluates the extend to which it is synchronized with the environment. This is measured by the fraction of time during which the robot finds tasks in its current workspace—see Sect 3.1. Second, each robot i modifies its internal timer τ^i and period T^i to synchronize with the environment—see Sect. 3.2 and 3.3. A robot i is synchronized with the environment when $T^i = T^{env}$ and $\bar{\tau}^i = 0$. Additionally, CS features a visual communication protocol to avoid physical interference between robots—again, see Sect. 3.2.

3.1 Assessment of Synchronization

Each robot i assesses its synchronization with the environment by measuring the correlation between its internal timer and the appearance of tasks. Robot i switches between workspaces every $T^i/2$, where T^i is updated by CS and is therefore not constant. Let l^i be a sequential number that identifies switches between workspaces for robot i, and let k_{l^i} be the moment in time at which switch l^i happens. Let $W^i[l^i]$ be the amount of time spent in a workspace by robot i between switch $l^i - 1$ and l^i, as opposed to transitioning between workspaces. See Fig. 2a-II.

Robot i can perform a task when it is in a workspace that contains available tasks. Let $w^i[k]$ a signal that takes the value 1 if robot i is working on a task at instant k and 0 if it is not. See Fig. 2a-III.

In order to assess its synchronization with the environment, robot i should ideally compute the correlation between the signal $s^i[k]$ of its internal timer and the signal $s^{env}[k]$ of the actual appearance of tasks:

$$g^i[k] = \begin{cases} 1 & \text{if } s^{env}[k] = s^i[k] \\ 0 & \text{if } s^{env}[k] \neq s^i[k] \end{cases} \quad (2)$$

$g^i[k]$ can be integrated over a time interval yielding a cross-correlation by which robots can evaluate the similarity of the two signals during the chosen interval. Let $r^i[l^i]$ define the cross-correlation between these signals during $W^i[l^i]$:

$$r^i[l^i] = \frac{1}{W^i[l^i]} \sum_{\kappa=0}^{W^i[l^i]} g^i\left[k_{l^i} - \kappa\right] \quad (3)$$

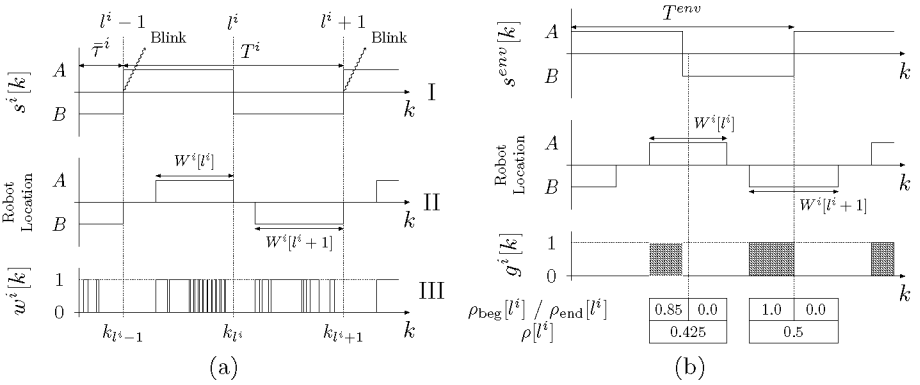

Fig. 2. Example of robot operation and cross-correlation of signals. a) Robot operation: I) signal $s^i[k]$ of the internal timer τ^i of robot i; II) robot location in the environment and amount of time W^i spent in a workspace (as opposed to transitioning between workspaces); III) work signal $w^i[k]$. b) Cross-correlation and partial cross-correlation, from top to bottom: signal $s^{env}[k]$ of task appearance; actual robot location; correlation of $s^{env}[k]$ and $s^i[k]$.

If robot i is perfectly synchronized with the environment, the two signals are identical; that is, $r^i[l^i] = 1$, $\forall l^i$. Typically, the two signals are not identical and $r^i[l^i]$ takes values lower than 1 for any value of l^i.

Unfortunately, $r^i[l^i]$ can not be directly measured because $s^{env}[k]$ is not known by the robots. Nevertheless, we can approximate $r^i[l^i]$ using the work signal $w^i[k]$: if robot i is performing a task, its internal timer and the location of task appearance match. Hence, we assume $w^i[k] \approx g^i[k]$. Let $\rho^i[l^i]$ define the cross-correlation between $s^i[k]$ and $w^i[k]$ during $W^i[l^i]$:

$$\rho^i[l^i] = \frac{1}{W^i[l^i]} \sum_{\kappa=0}^{W^i[l^i]} w^i\left[k_{l^i} - \kappa\right] \tag{4}$$

Contrarily to $r^i[l^i]$, $\rho^i[l^i]$ can be measured by robot i and provides it with an estimated assessment of the synchronization between its internal timer and task appearance.

3.2 Synchronization of the Internal Timer

To achieve internal timer synchronization with $s^{env}[k]$, robot i uses the cross-correlation $\rho^i[l^i]$ defined in (4). We additionally define two partial cross-correlations:

$$\begin{aligned} \rho^i_{beg}[l^i] &= \frac{2}{W^i[l^i]} \sum_{\kappa=W^i[l^i]/2}^{W^i[l^i]} w^i\left[k_{l^i} - \kappa\right] \\ \rho^i_{end}[l^i] &= \frac{2}{W^i[l^i]} \sum_{\kappa=0}^{W^i[l^i]/2} w^i\left[k_{l^i} - \kappa\right] \end{aligned} \tag{5}$$

where $\rho^i_{beg}[l^i]$ is computed for the first half of $W^i[l^i]$ and $\rho^i_{end}[l^i]$ for the second. The comparison between these two quantities measures the balance of work between the two halves of $W^i[l^i]$: $\rho^i_{beg} \neq \rho^i_{end}$ indicates that robot i is working more during one half of $W^i[l^i]$ than the other. See Fig. 2b.

Robot i uses the relationship between ρ^i_{beg} and ρ^i_{end} to shift its internal timer as follows:

$$\Delta\tau^i[l^i] = \frac{W^i[l^i]}{2}\left(\rho^i_{beg}[l^i] - \rho^i_{end}[l^i]\right) \tag{6}$$

where $\Delta\tau^i[l^i]$ denotes the timer modifier of robot i at switch l^i, which occurs at the end of $W^i[l^i]$.

Collective Synchronization: Additionally to (6), we propose a mechanism for collective synchronization that is based on short-range line-of-sight communication. Communication is implemented using a simple visual protocol: a robot emits a light blink when its internal timer has finished a full cycle, thereby signaling to other robots that its timer is zero—see Fig. 2a-I. Other robots perceiving this light blink adjust their timers to achieve collective synchronization.

Upon perceiving a light blink, robot i shifts its internal timer as follows:

$$\Delta\tau^i = \begin{cases} 0.1\left(\beta T^i - \tau^i\right) & \text{if } \tau^i \leq T^i/2 \\ 0.1\left(T^i - \beta T^i - \tau^i\right) & \text{if } \tau^i > T^i/2 \end{cases} \tag{7}$$

where $\beta \in [0, 0.5)$ is a parameter.

In case $\beta T^i < \tau^i < T^i - \beta T^i$, robot i shifts its timer such that its reset point is closer to one of the emitting robot. This provokes a coupling and clustering of the timers. On the level of the swarm, the cluster of timers tends to synchronize with $s^{env}[k]$ because each timer is modified by (6). On the other hand, if $\tau^i < \beta T^i$ or $\tau^i > T^i - \beta T^i$, robot i shifts its timer so that its reset point is farther from the one of the emitting robot. This avoids that timers are too closely clustered, which would cause all robots to cross the transition area at the same time, thereby creating physical interference. Notice that in (7) there is no reference to an absolute time as robots do not share a common time reference and the visual communication protocol is asynchronous.

3.3 Period Synchronization

To achieve period synchronization of signals $s^i[k]$ and $s^{env}[k]$, robot i uses two statistics of the cross-correlation $\rho^i[l^i]$: the exponential moving average $avg\left(\rho^i[l^i]\right)$ and the variance $var\left(\rho^i[l^i]\right)$. These statistics are updated with the current value of the cross-correlation $\rho^i[l^i]$, using a memory factor η:

$$\begin{aligned} avg\left(\rho^i[l^i]\right) &= \eta\ avg\left(\rho^i[l^i - 1]\right) + (1-\eta)\rho^i[l^i] \\ var\left(\rho^i[l^i]\right) &= \eta\ var\left(\rho^i[l^i - 1]\right) + (1-\eta)\left(\rho^i[l^i] - avg\left(\rho^i[l^i]\right)\right)^2 \end{aligned} \tag{8}$$

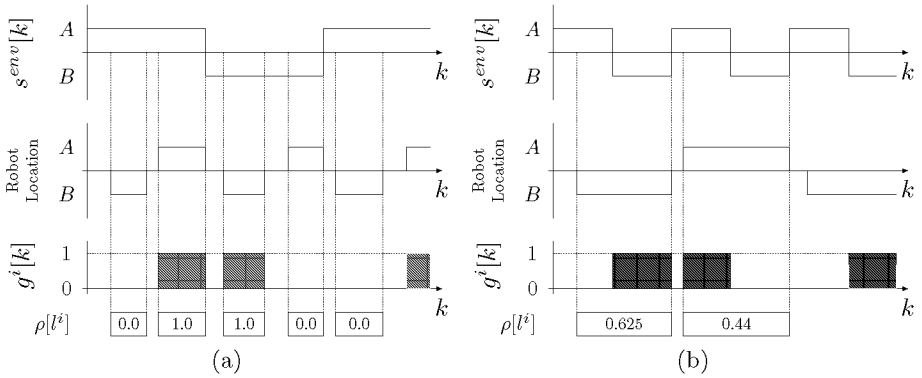

Fig. 3. Cross-correlation examples for a) $T^i < T^{env}$, where $avg\left(\rho^i[l^i]\right) \rightarrow 0.5$ and $var\left(\rho^i[l^i]\right) \rightarrow 0.25$ and b) $T^i > T^{env}$, where $avg\left(\rho^i[l^i]\right) \rightarrow 0.53$ and $var\left(\rho^i[l^i]\right) \rightarrow 0.008$. From top to bottom: location of task appearance ($s^{env}[k]$), robot location and cross-correlation.

with $\eta \in [0, 1]$ being a configurable parameter. The higher η, the more relevant the current value. $avg\left(\rho^i[l^i]\right)$ measures the difference between T^i and T^{env}: the closer $avg\left(\rho^i[l^i]\right)$ to 1, the smaller the difference is. As $avg\left(\rho^i[l^i]\right)$ only indicates a difference, but not if T^i is shorter or longer, we use $var\left(\rho^i[l^i]\right)$ to measure the length of T^i in relation to T^{env}: if T^i is shorter than T^{env}, $var\left(\rho^i[l^i]\right)$ goes to 1. Figure Fig. 3 illustrates these statistics and their values in two example situations.

Robot i modifies the period of its internal timer $s^i[k]$ as follows:

$$\Delta T^i[l^i] = W^i[l^i]\left(f_{inc}\left(var\left(\rho^i[l^i]\right)\right) - f_{dec}\left(avg\left(\rho^i[l^i]\right)\right)\right) \tag{9}$$

where $f_{inc}\left(var\left(\rho^i[l^i]\right)\right)$ and $f_{dec}\left(avg\left(\rho^i[l^i]\right)\right)$ are positive functions that increase and decrease the period, respectively.

We use the following function $f_{inc}(var\left(\rho^i[l^i]\right))$ for increasing the period:

$$f_{inc}\left(var\left(\rho^i[l^i]\right)\right) = W^i[l^i]\alpha_{var}\left(var\left(\rho^i[l^i]\right)\right)^2 \tag{10}$$

where $\alpha_{var} \in \mathbb{R}$ is a configurable parameter that regulates the influence of $var\left(\rho^i[l^i]\right)$ on the period. We use the following function $f_{dec}\left(avg\left(\rho^i[l^i]\right)\right)$ for decreasing the period in this paper:

$$f_{dec}\left(avg\left(\rho^i[l^i]\right)\right) = W^i[l^i]\alpha_{avg}\left(1 - avg\left(\rho^i[l^i]\right)\right)^2 \tag{11}$$

where $\alpha_{avg} \in \mathbb{R}$ is a configurable parameter that regulates the influence of $avg\left(\rho^i[l^i]\right)$ on the period.

4 Experiments

We conduct the experiments in simulation using ARGoS [13]. ARGoS is a discrete-time physics-based simulation framework whose focus is the simulation

of large robot swarms. The arena that we use in the experiments has the same layout as shown in Fig. 1a, with a length of 120 cm, a width of 60 cm, and a workspace width of 30 cm.

For our experiments, we use a swarm of 6 *e-puck* robots [12], which are randomly distributed in the transition area upon the start of the experiments. We use the following sensors of the e-puck: proximity sensors for obstacle avoidance, ground sensors to detect floor color, light sensor for phototaxis and the camera for task detection and visual communication. We use the wheel actuator with a maximum speed of $8\,^{\mathrm{cm}}/_{\mathrm{s}}$. Additionally, we use the LED actuator to implement the visual communication protocol.

We represent tasks using a device called *task allocation module* (TAM) [3]. A TAM represents a task to be executed by an e-puck robot at a given location and at a given moment in time. TAMs are programmable booths that signal the availability of a task to the robots through a set of color LEDs. A robot can work on the task that is represented by a TAM by driving into it and waiting inside until ξ_e has elapsed. We placed 10 TAMs in each workspace; hence, the task capacity of each workspace is $\Gamma = 10$. The time that a robot needs to perform a task is $\xi_e = 0.5\,\mathrm{s}$. The task life time is $\xi_l = 5\,\mathrm{s}$.

Robots use phototaxis to navigate: a light source identifies the right side of the arena. Robots navigate towards the light to work in workspace B, and do the opposite to work in workspace A. Robots can detect the workspace they are in by reading the color of the floor. When a robot is in a workspace, it perceives the available tasks by the color of the LEDs of the nearby TAMs. Robot i can calculate $W^i[j]$ by measuring the time at which it arrives in a workspace and the time at which the internal timer switches to the other workspace. The robot can sense the work signal $w^i[k]$ by the color of the TAMs in its current workspace.

The parameters of the environment are the incoming task rate $\lambda = 10\,^{\mathrm{tasks}}/_{\mathrm{s}}$ and the period $T^{env} = 80\,\mathrm{s}$. The configurable parameters of CS used for this example are $\eta = 0.65$, $\alpha_{avg} = 0.58$ and $\alpha_{var} = 41.92$. These values have been obtained through a tuning process using I/F-Race [1,10]. The parameter $\beta = 0.0375$ has been obtained by exhaustive search. Initially, the period T^i of each robot i is uniformly sampled from the interval $[40, 240]$, and the initial time difference between the internal timers and the task appearance $\bar{\tau}^i$ is uniformly sampled from the interval $[-T^i/2, T^i/2]$.

Figure 4 shows the development of T^i, $\bar{\tau}^i$ and number of tasks performed over the duration of the experiment. The synchronization of the periods in the swarm is shown in Fig. 4a. Notice that T^{env} is constant during the experiment. We can observe that every T^i converges to T^{env} in the first 2500 s. The timer synchronization is shown in Fig. 4b. We can observe that all $\bar{\tau}^i$ converge to zero. The number of tasks performed by the swarm during the experiment is a cumulative metric shown in Fig. 4c. We define the *performance rate* as the number of tasks performed per second which is the derivate of this metric. In this example, the robots achieve a performance rate of $0.65\,^{\mathrm{tasks}}/_{\mathrm{s}}$.

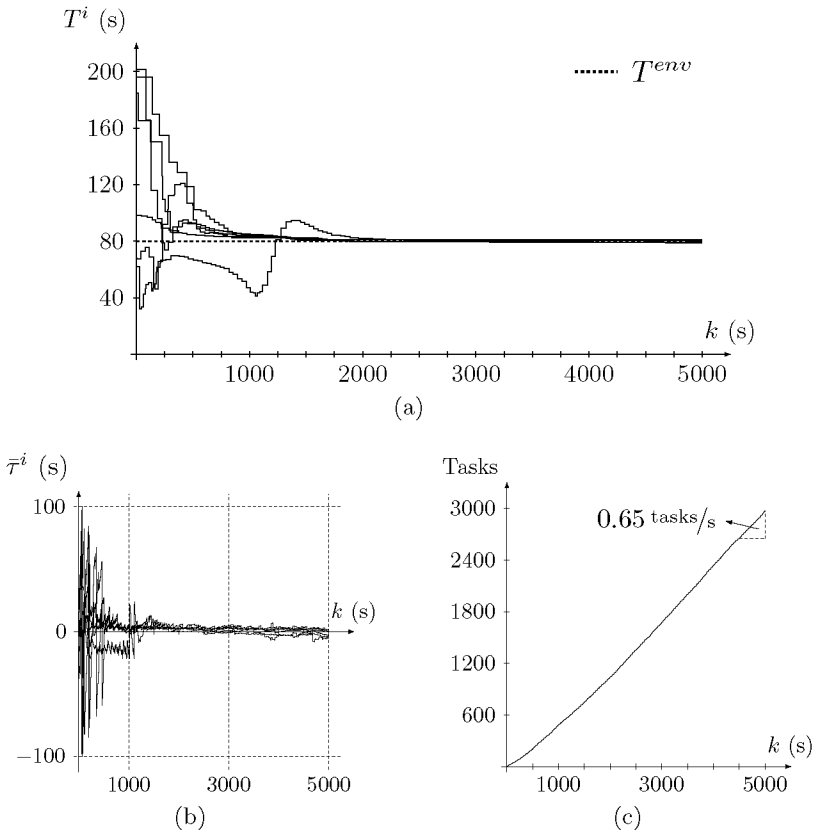

Fig. 4. Development of T^i, $\bar{\tau}^i$ and number of tasks performed over the duration of the experiment. a) Periods T^i of the robots compared to the period T^{env}. b) Time difference $\bar{\tau}^i$ between the internal timers and the task appearance. c) Number of tasks performed and final performance rate.

In order to analyze the performance of CS, we compare it with two other algorithms:

- No-synchronization algorithm (NS): robots using NS have internal timers for switching between areas, but do not attempt to synchronize with the environment or with other robots. This means that $\bar{\tau}^i$ and T^i, which are randomly initialized, remain unchanged throughout the experiment. This represents the initial situation of CS. The comparison of CS with NS allows us to quantify the improvement obtained by synchronizing.
- Greedy algorithm (GR): robots using GR do not switch between workspaces depending on an internal timer, but on task availability. To this end, robots switch workspace with a given probability in case they to not find tasks in their current workspace. The comparison of CS with GR allows us to observe the difference in task performance between a temporal task allocation algorithm and an algorithm that is commonly used in task allocation.

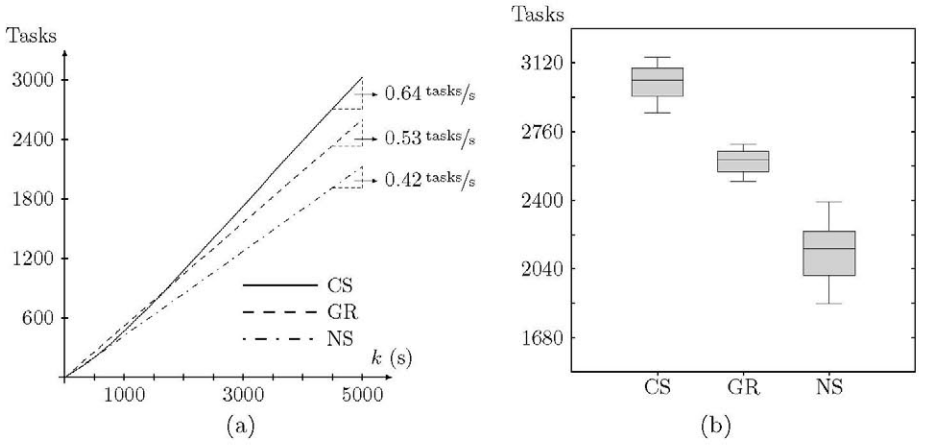

Fig. 5. Comparison of the performance of CS, NS and GR. a) Average of the number of tasks performed during the experiments; and final performance rate. b) Boxplot of the number of task performed by each algorithm during the experiment, based on 15 repetitions per algorithm. The whiskers represent the lowest value still within 1.5 IQR of the lower quartile, and the highest value still within 1.5 IQR of the upper quartile.

The comparison between algorithms is based on the number of tasks performed in experiments of 5000 s. We perform 15 experiments for each algorithm.

The average number of tasks performed by the swarm, calculated every second for each algorithm, is shown in Fig. 5a. Notice that the performance rates of NS and GR remain constant during the experiment because these algorithms do not implement a synchronization process. We can observe that GR has a higher performance rate than NS. This implies that a temporal task allocation algorithm that is not perfectly synchronized performs less tasks than a greedy algorithm. Furthermore, we can observe that the performance rate of CS increases over time due to the synchronization process. At the beginning of the experiments, CS and NS have a similar performance rate. After 800 s, the performance rate of CS increases and eventually the number of tasks performed by CS exceeds the number of tasks performed by NS. Similarly, the number of tasks performed by CS exceeds the number of tasks performed by GR after 1600 s.

Figure 5b shows a boxplot representation of the number of task performed at the end of the experiments. The NS algorithm has the highest dispersion because there is no synchronization; the results strongly depend on the initial conditions. CS and GR have a lower dispersion than NS as they adapt the behavior of the robots to the environment, reducing the dependence on the initial conditions. Notice that the lower whisker of CS is longer than the upper whisker of the GR algorithm. This implies that the previous deductions made on Fig. 5a are valid.

5 Conclusions

In this paper, we studied task allocation in an environment that exhibits periodic temporal patterns. In such an environment, robots can perform temporal task allocation to exploit synchronization for improving their task performance. We have described and analyzed a collective synchronization algorithm that performs temporal task allocation for robot swarms. In order to analyze the performance of the proposed algorithm, we compared it with a no-synchronization algorithm and a greedy algorithm. From the results, we can conclude that a swarm using our algorithm can synchronize with the environment, thereby outperforming the competing algorithms. The comparison also shows that a temporal task allocation algorithm without synchronization performs less tasks than a greedy algorithm. However, an algorithm with synchronization as proposed by us increases the number of tasks performed by the robots considerably with respect to a reactive behavior.

In this paper, we applied the concepts of synchronization and signal processing to task allocation in swarm robotics, with satisfactory results. In the immediate future, we plan to study the proposed approach on a swarm of real robots. The potential of our approach opens several possible directions for future research. One is to study environments that exhibit more complex temporal patterns, for example, environments in which the task appearance is not only a square signal but a signal with multiple frequency components. Another direction is the application of this approach to other resource allocation problems such as energy management. For example, they can be used to organize the consumption of a scarce energy resource by a swarm of robots or other autonomous agents such as electric vehicles.

Acknowledgments. M. Castillo-Cagigal is sponsored by the Spanish Ministry of Education with a PhD grant (FPU-2010). Arne Brutschy and Mauro Birattari acknowledge support from the Belgian F.R.S.–FNRS.

References

1. Balaprakash, P., Birattari, M., Stützle, T.: Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In: Bartz-Beielstein, T., Blesa Aguilera, M.J., Blum, C., Naujoks, B., Roli, A., Rudolph, G., Sampels, M. (eds.) HCI/ICCV 2007. LNCS, vol. 4771, pp. 108–122. Springer, Heidelberg (2007)
2. Brambilla, M., Ferrante, E., Birattari, M., Dorigo, M.: Swarm robotics: A review from the swarm engineering perspective. Swarm Intelligence 7(1), 1–41 (2013)
3. Brutschy, A., Garattoni, L., Brambilla, M., Francesca, G., Pini, G., Dorigo, M., Birattari, M.: The TAM: abstracting complex tasks in swarm robotics research. Tech. Rep. TR/IRIDIA/2014-006, IRIDIA, Université Libre de Bruxelles, Belgium (2014)
4. Christensen, A.L., O'Grady, R., Dorigo, M.: From fireflies to fault-tolerant swarm of robots. IEEE Transactions on Evolutionary Computation 13(4), 754–766 (2009)

5. Dorigo, M., Birattari, M., Brambilla, M.: Swarm robotics. Scholarpedia 9(1), 1463 (2014)
6. Glass, L.: Synchronization and rhythmic processes in physiology. Nature 410(6825), 277–284 (2001)
7. Holland, O., Melhuish, C., Hoddell, S.E.J.: Convoying: using chorusing for the formation of travelling groups of minimal agents. Robotics and Autonomous Systems 28, 207–216 (1999)
8. Kuramoto, Y.: Chemical Oscillations, Waves, and Turbulence. Springer, Berlin (1984)
9. Liu, F., Picard, R.W.: Finding periodicity in space and time. In: Proceedings of the 6th International Conference on Computer Vision, pp. 376–383. IEEE Computer Society, Los Alamitos (1998)
10. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The irace package, iterated race for automatic algorithm configuration. Tech. Rep. TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011)
11. Martin, H.J.A., de Lope, J., Maravall, D.: Adaptation, anticipation and rationality in natural and artificial systems: computational paradigms mimicking nature. Natural Computing 8(4), 757–775 (2009)
12. Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.C., Floreano, D., Martinoli, A.: The e-puck, a robot designed for education in engineering. In: Gonçalves, P.J.S., et al. (eds.) Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, pp. 59–65. IPCB: Instituto Politècnico de Castelo Branco, Portugal (2009)
13. Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L.M., Dorigo, M.: ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. Swarm Intelligence 6(4), 271–295 (2012)
14. Rosu, D., Schwan, K., Yalamanchili, S., Jha, R.: On adaptive resource allocation for complex real time applications. In: Proceedings of the 18th Real-Time System Symposium, pp. 320–329. IEEE Computer Society, Los Alamitos (1997)
15. Strogatz, S.H.: From Kuramoto to Crawford: exploring the onset of synchronization in populations of coupled oscillators. Physica D: Nonlinear Phenomena 143(1-4), 1–20 (2000)